

# GhostFill vs. HotDocs



**Seth G. Rowland, Esq.**  
CEO & President  
Basha Systems LLC  
[sgr@bashasys.com](mailto:sgr@bashasys.com)  
[www.bashasys.com](http://www.bashasys.com)





# Agenda

- Living with HotDocs
- Head to Head
  - Developer Experience
  - End User Experience





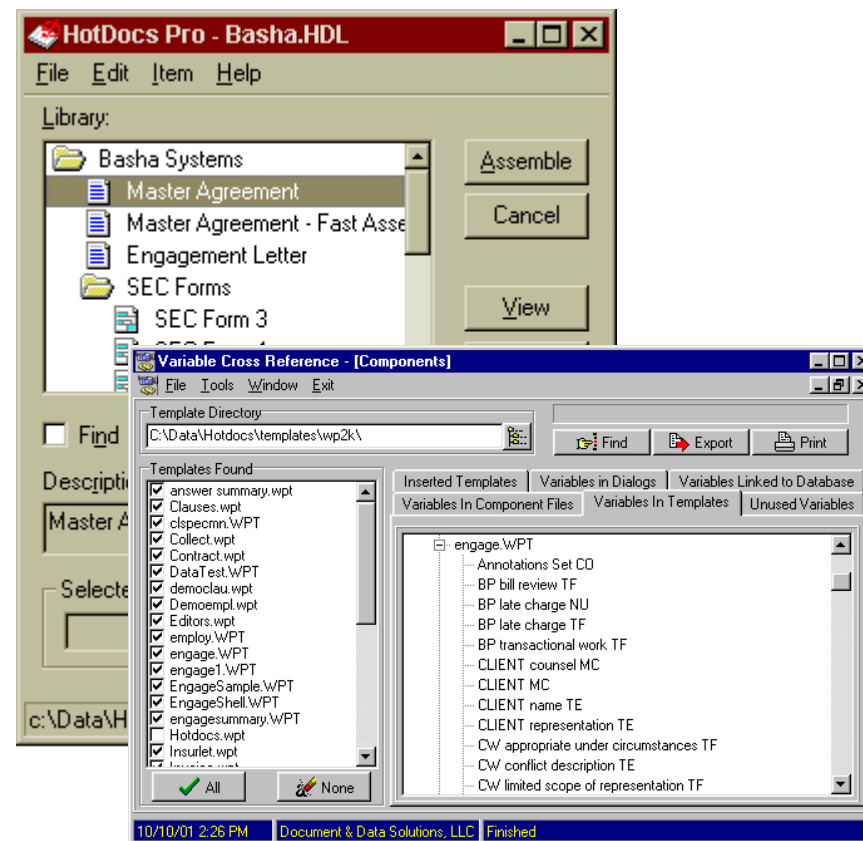


# Head to Head

## *Library vs. Application*



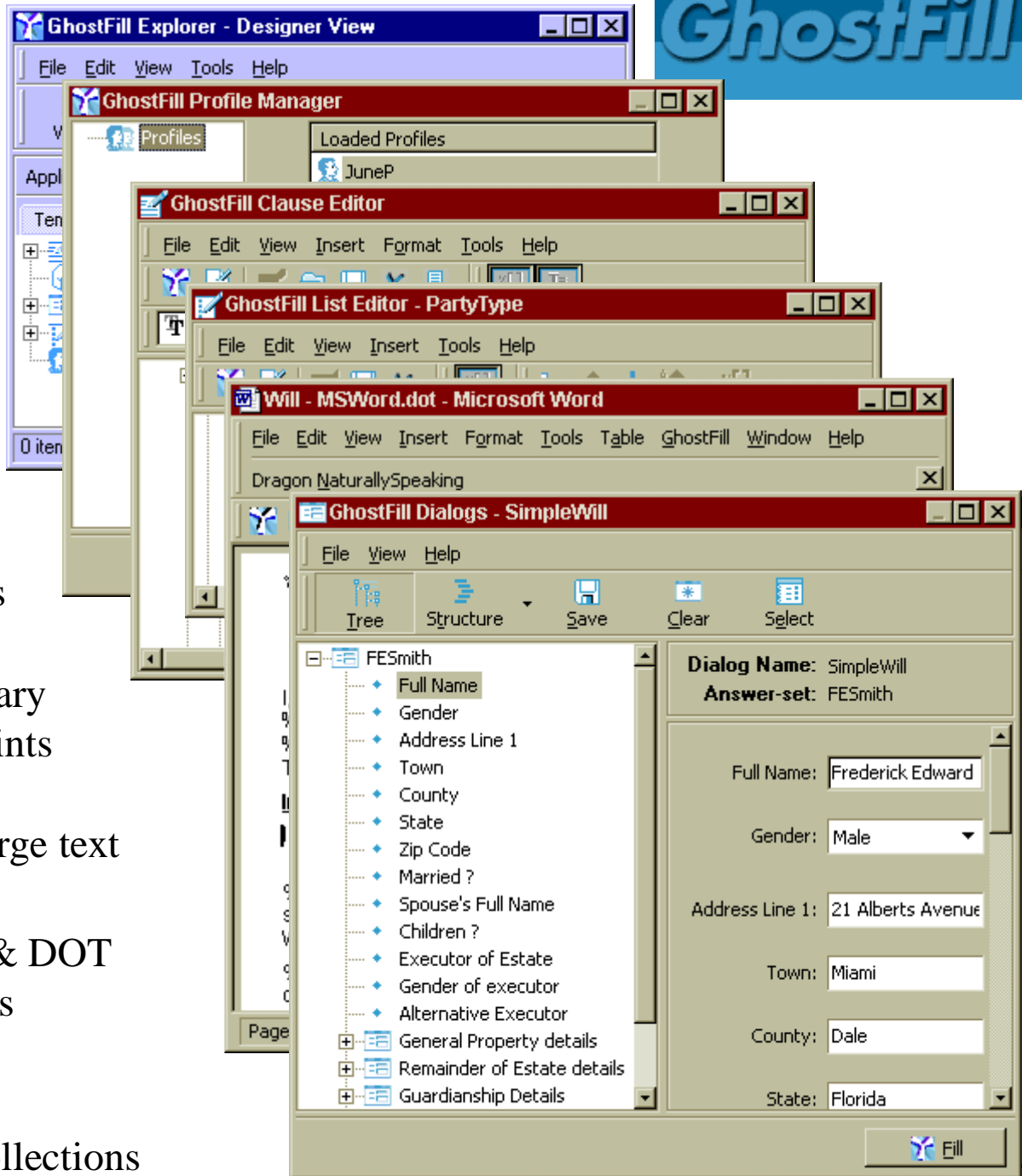
- Basic HotDocs Library
  - Individual libraries (local)
  - Network libraries (read only)
  - Developer libraries
- Hybrids and Customization
  - Use of Reference Paths
  - Local library with embedded master library
- Templates
- Component Files
  - Pointing vs. Replication
  - Even with DDS's VCR Utility





## Head to Head *Library vs. Application*

- GF Explorer
  - User View
  - Designer View
- GF Profile Manager
  - Multiple Profiles
  - Live modifications
- GF Clause Editor
  - Classic clause library
  - Support for Fillpoints
- GF List Editor
  - Programmable merge text
- GF Templates
  - RTF, HTM, TXT & DOT
  - Power of Fillpoints
- GF Dialogs
  - Tree and Details
  - Sub-Dialogs & Collections



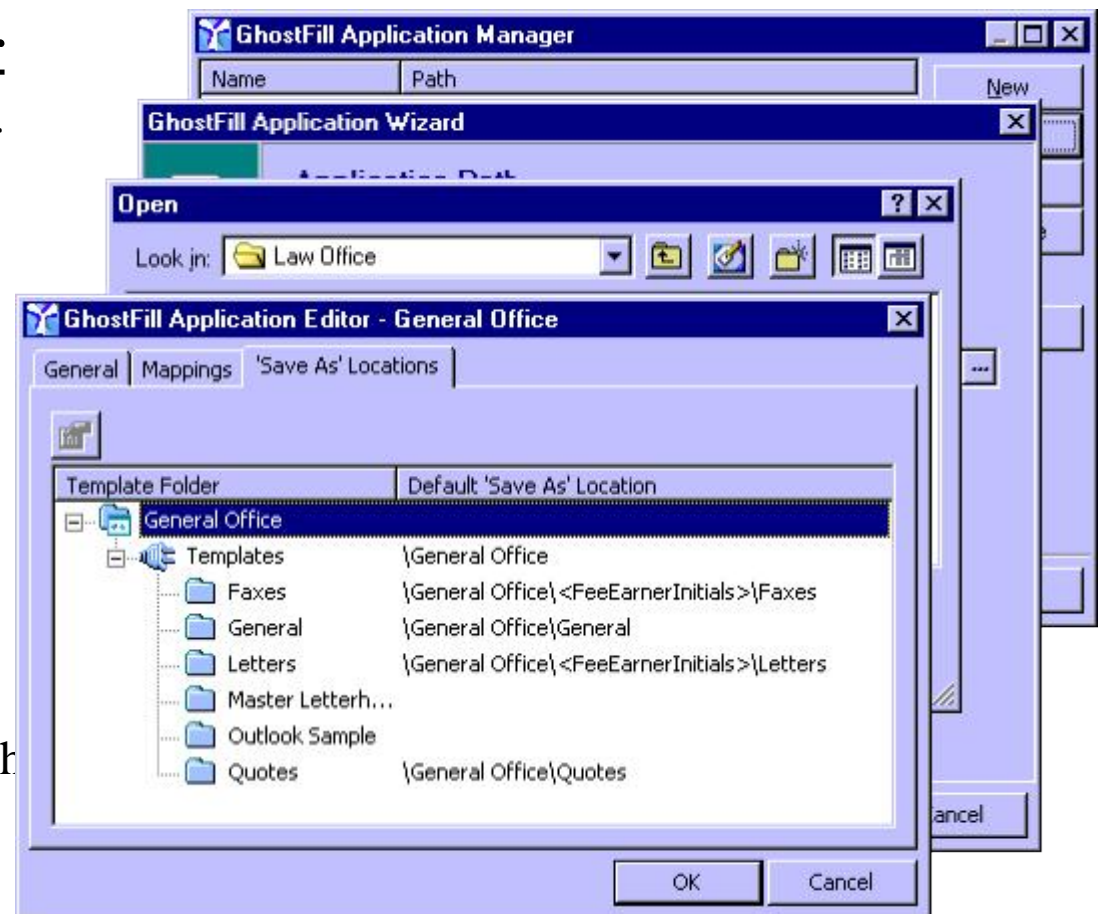


# Head to Head

## *Library vs. Application*

### GF Application Manager

- GF Application Wizard for new applications
- Just tell GF where to create the application
- To import GF Application, find the XML definition file
- To revise or configure and application – basic info
- Revise file mappings – multiple location support
- Set ‘Save As’ Location with profile variable support



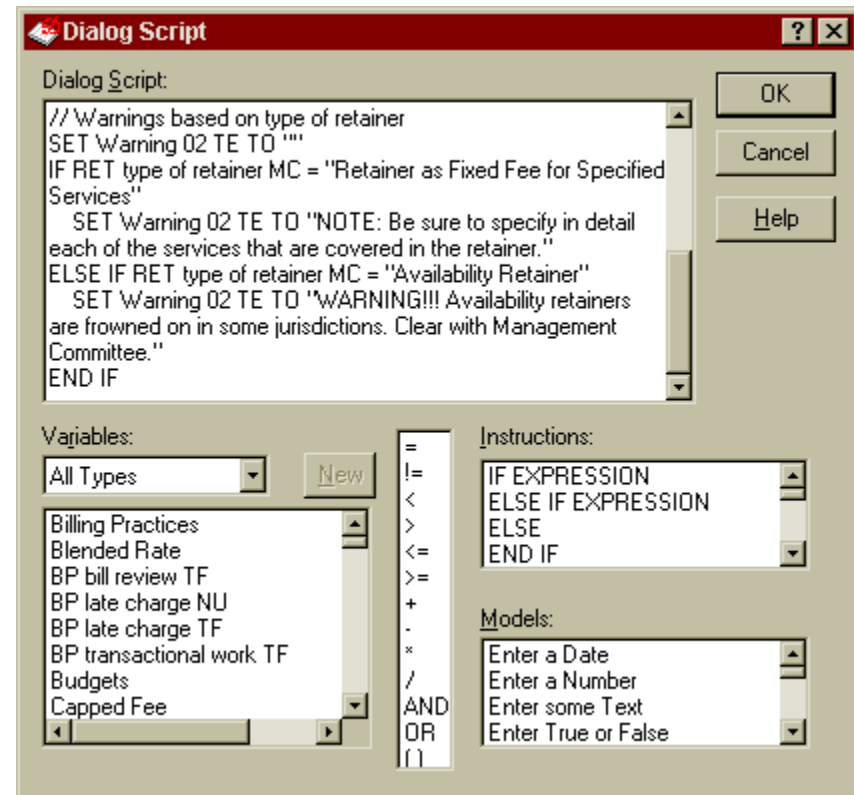


# Head to Head

## Building Dialogs



- A customized dialog built with HotDocs Pro
- TEST button to view dialog as you build it
- EDIT button and ability to relocate variables
- Variables script – Use of GRAY and UNGRAY
- Warning script – use of the SET TO command



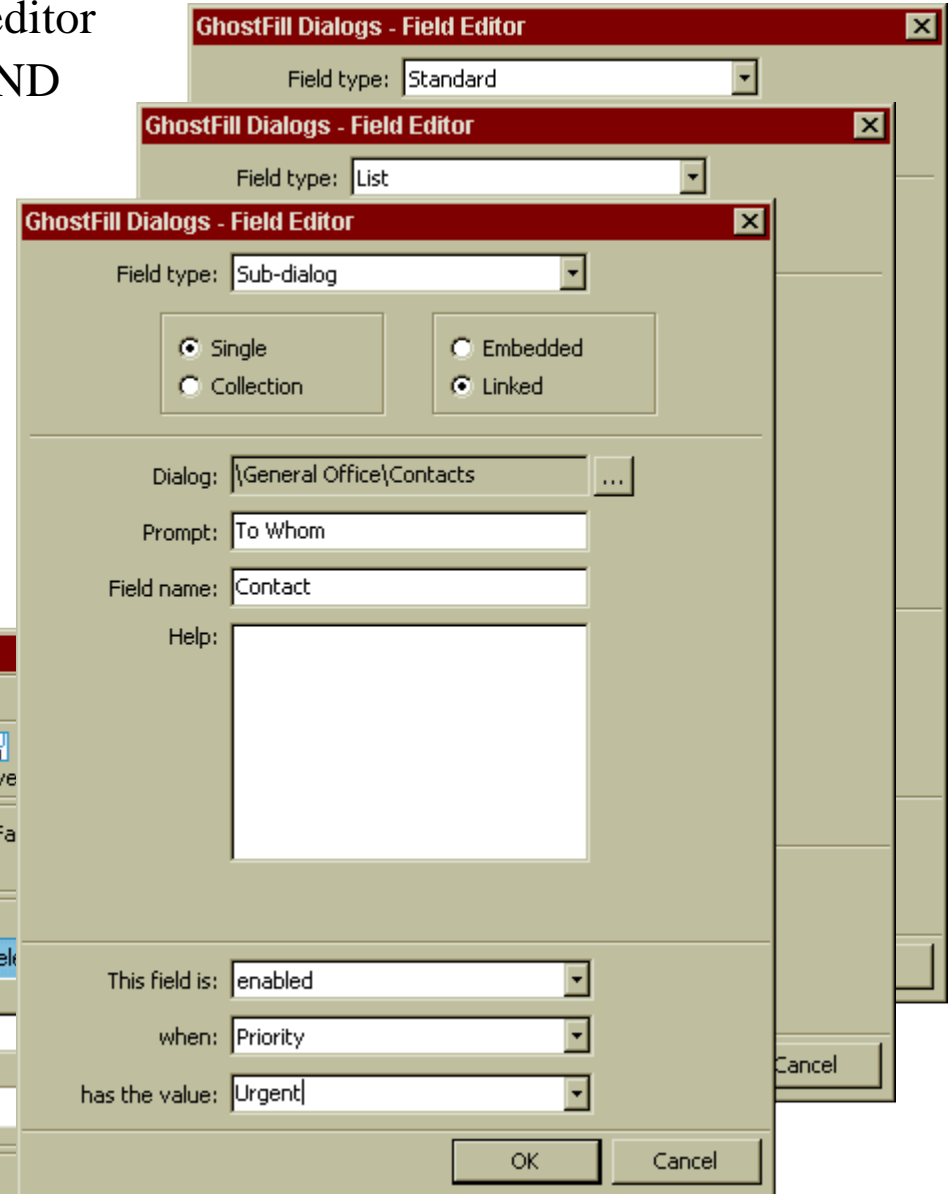
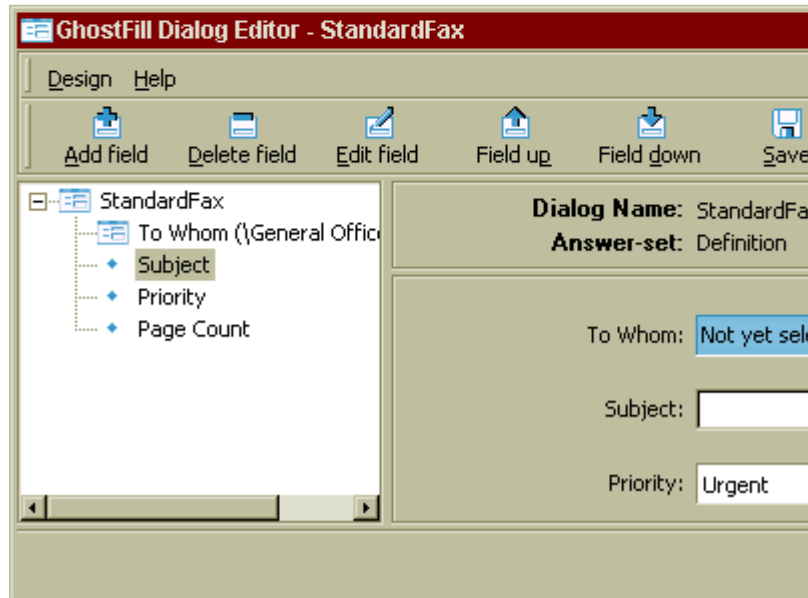


# Head to Head

## Building Dialogs



- Choice of dialog wizard or dialog editor
- Support for text, TF, dates, time, AND sub-dialogs, picklists & arrays
- Variables are integrated into & defined by the dialog
- Great control over lists, sub-dialogs and collections
- Set conditions without scripting
- Advanced scripting, including COM calls







# Head to Head

## *Fillpoints and Tasks*



### Types of Fillpoints

- Basic variable
- Text formatting
- Date formatting
- Alternate text
- Basic IF statement
- Delete Block
- Get clause block
- Dialog calls
- Repeats

«WILL Spouse TE»

«WILL Benefactor TE:LIKE THIS»

«WILL Date of Agreement DA:9 June 1990»

«WILL Benefactor gender TE:he/she»

«IF WILL Married TF» ... «END IF»

«IF UNANSWERED(WILL Guardian TE) != FALSE»

«INSERT IF WILL Stepchildren CL»

«ASK Children DLG»

«REPEAT Children DLG»



# Head to Head

## *Fillpoints and Tasks*



### Types of Fillpoints

- Basic variable
- Text formatting
- Date formatting
- Alternate text
- Basic IF statement
- Delete Block
- Get clause block
- Dialog calls
- Repeats

`%[Will.Spouse]`

`%[UpperCase(Will.Benefactor)]`

`%[DateToWords(DateOfAgreement, False)]`

`%[Iftrue(Will.gender="Male", "his", "her")]`

`%[KeepBlockIf(Will.Married)] .... %[EndBlock()]`

`%[DeleteBlockIf(Will.Guardian.Name="")]`

`%[Clauses.Select("\nda\", "Parties Description")]`

`%[^Kids = Dialogs.Capture("\Office\", "", False)]`

`%[RepeatBlockWhile(Index<Kids.Count)]`



# Head to Head

## *Fillpoints & Tasks*

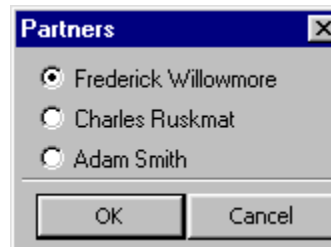


### Lists & Profiles

- Define the list
- Call up the list (single select)
- Choose the Partner
- Results defines name, initials and signature
- Select clause uses the result
- Pull in signature
- Define signatory
- Result sets fullname

```
%[P = Lists.Select("\Office\Firm","Partners")]
```

```
%[P.SingleSelectList("Partners", 1)]
```



```
%[Fullname="Frederick  
John Willowmore";  
Initials="FJW";  
Signature="FJWillowmore"]
```

```
%[Clauses.Select("\Office\Sign\","Signature)]
```

*Fred Willowmore*  
WILLOWMORE RUSKMAT & SMITH

```
%[FullName]
```

**Frederick John Willowmore**



# Head to Head

## *Fillpoints and Tasks*



### Working with Objects: Gender

- Define the Object
- Use the object, with proper syntax
- Apply to any variable

```
%[Gender=CreateObject("Legal.Litigation")]
```

```
Apostrophe: %[Gender.Parties("Party").sapp]
```

```
Does/Do: %[Gender.Parties("Party").DoesDo]
```

```
ies/y: %[Gender.Parties("Party").iesY]
```

```
me/us: %[Gender.Parties("Party").MeUs]
```

```
has/have: %[Gender.Parties("Party").HasHave]
```

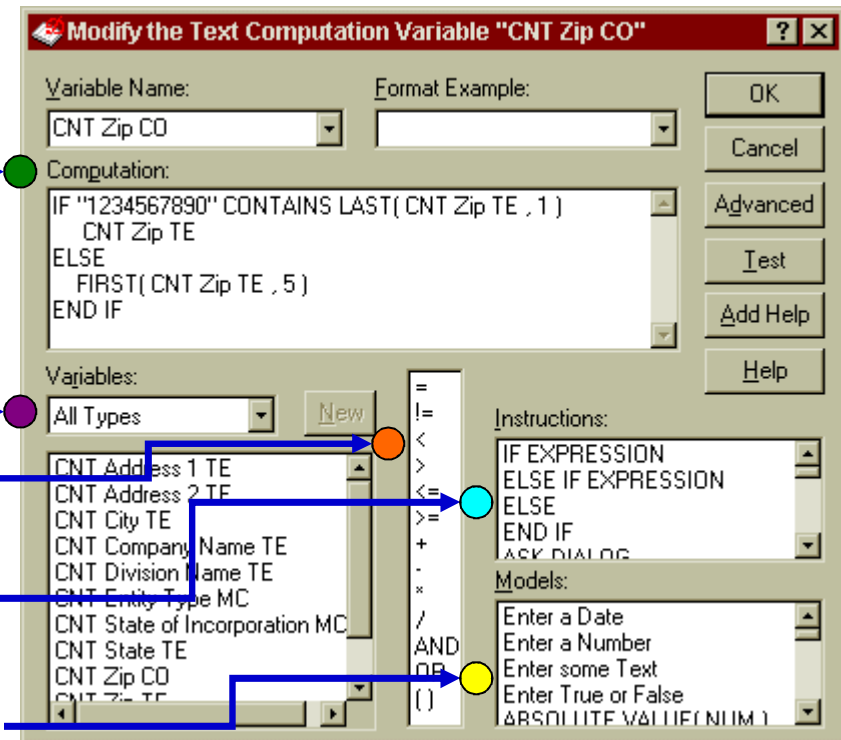


# Head to Head

## Computation Editor vs. Fillpoint Editor



- Computation Editor and Expression Editor
  - Mini-program with syntax the mirrors programming language
  - Access all variables, dialogs, filters, etc. or create one
  - Operators: =, !=, AND, ()
  - Instructions: IF, ELSE, ASK, REPEAT, FILTER
  - Models: COUNT, ANSWERED, INTEGER



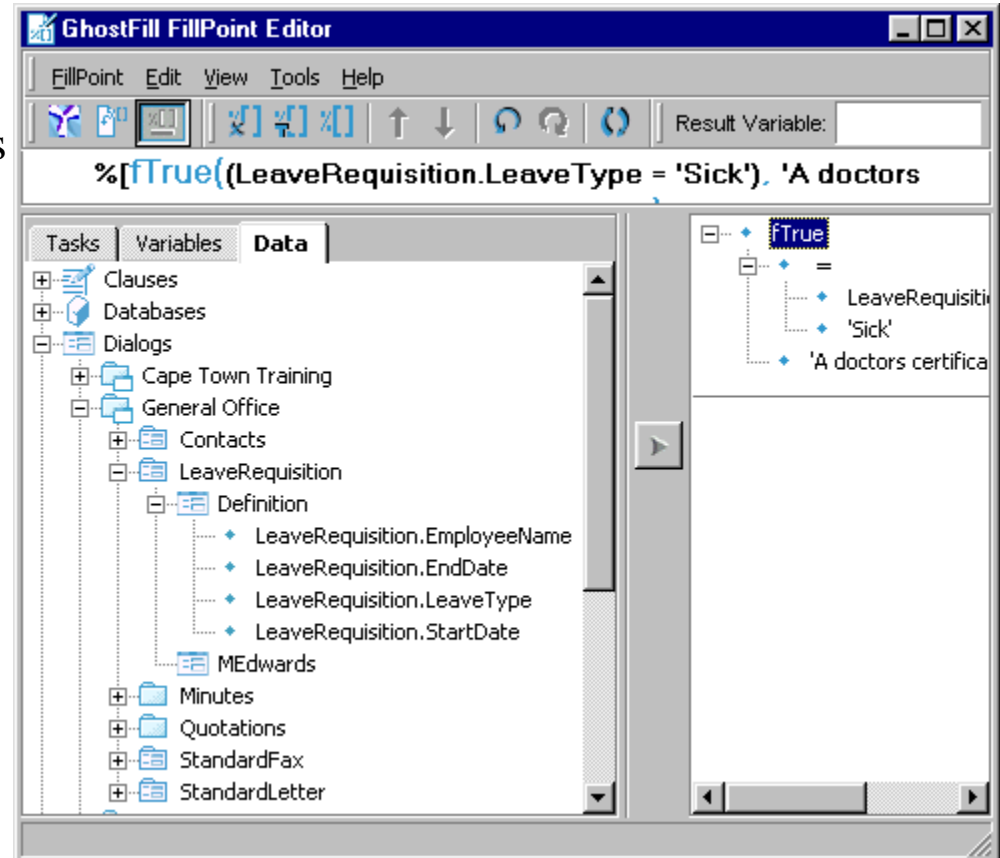


# Head to Head

## *Computation Editor vs. Fillpoint Editor*

- Basic Fillpoint
  - Simple to Complex
  - Built with Fillpoint Editor
- Tree Logic
  - Break into tasks and objects
  - Just click to expand
- Task List
  - Complete set of functions
  - Syntax builder and help
- Variables List
  - All open templates
- Data List
  - Clauses and Databases
  - Templates and Documents
  - Lists and Profiles

```
[IfTrue((LeaveRequisition.LeaveType = "Sick"), "A doctors certificate is attached.")]
```





# Head to Head

## External Data Sources



### Database Connection

#### *Defining the Source*

- Select ODBC from Windows Control Panel
- Click ADD in ODBC Data Source Administrator
- Select Driver for the new data source
- Configure Excel driver for the data source
- Select the data file

The image shows a sequence of Windows XP dialog boxes used to configure an ODBC data source for an Excel file:

- Control Panel**: The ODBC icon is highlighted in the Administrative Tools section.
- ODBC Data Source Administrator**: The 'User DSN' tab is active, and the 'Add...' button is clicked.
- Create New Data Source**: A list of drivers is shown, with 'Microsoft Excel Driver (\*.xls)' selected.
- ODBC Microsoft Excel Setup**: The 'Data Source Name' is set to 'HD Demo Excel'.
- Select Workbook**: The file 'ProskauerBillingRate.XLS' is selected in the file list.



# Head to Head

## External Data Sources



### Database Connection

#### *Create & Configure*

- Open Component Manager and create New database component
- Name the component
- Select the pre-defined data source and select the table
- Map the fields in the data source to predefined variables
- Select display fields
- Define the sort order
- Configure Ask Options
- Apply pre-defined filter

The screenshot displays the 'Component Manager' window with the 'Link to Database' dialog box open. The 'Database Component Name' is set to 'XLS staff link DB', the 'Data Source' is 'Proskauer Staff XLS', and the 'Table Name' is 'qryTimekeeperInfo'. A table below shows the mapping of fields:

Key	Field Name	Linked Variable
1	EmpId	XLS staff ID TE

The 'Select Display Fields' dialog shows 'EmpId' selected for display. The 'Sort' dialog shows 'group\_title' selected for sorting. The 'Filter Records' dialog shows three filters:

Field	Comparison	Compared To
Location	LIKE	XLS location filter T
AND DeptDesc	LIKE	XLS department filter
OR BillRate	GREATER THAN	XLS no filter TMP
AND		





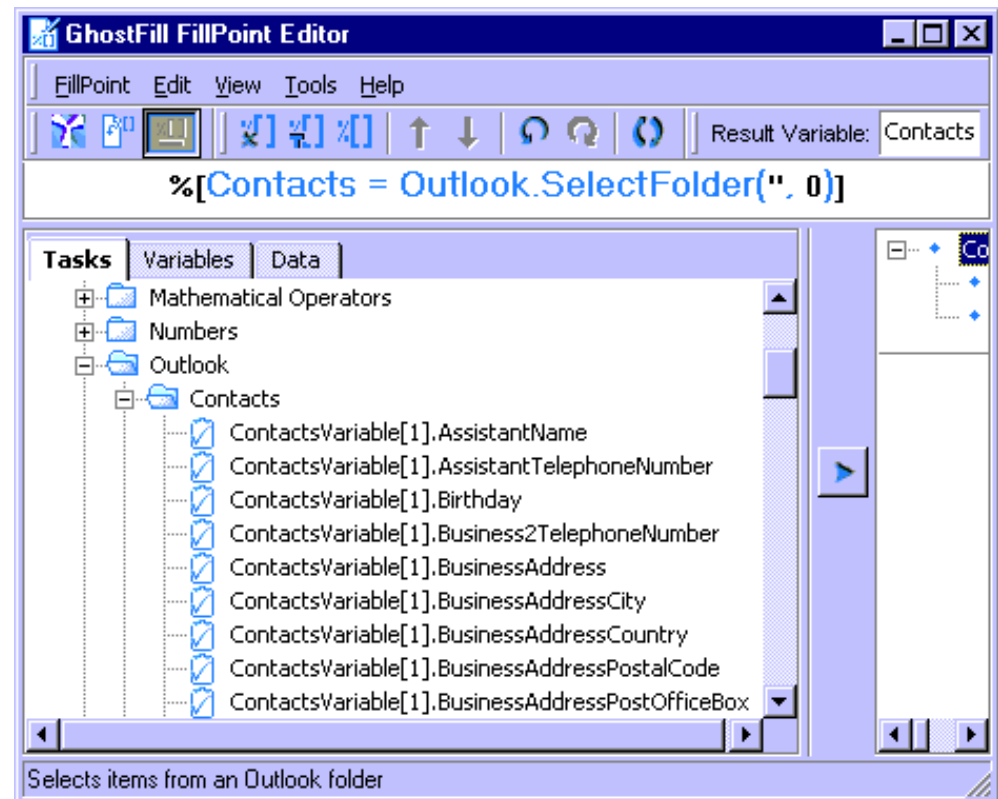
# Head to Head

## *External Data Sources*



### MS Outlook (COM)

- Define Outlook link
- Get contact name
- For multiple contacts, first set index to 1
- Repeat block with conditions, indexed variables and counter
- Conditions on salutation to use first name or “Sir”





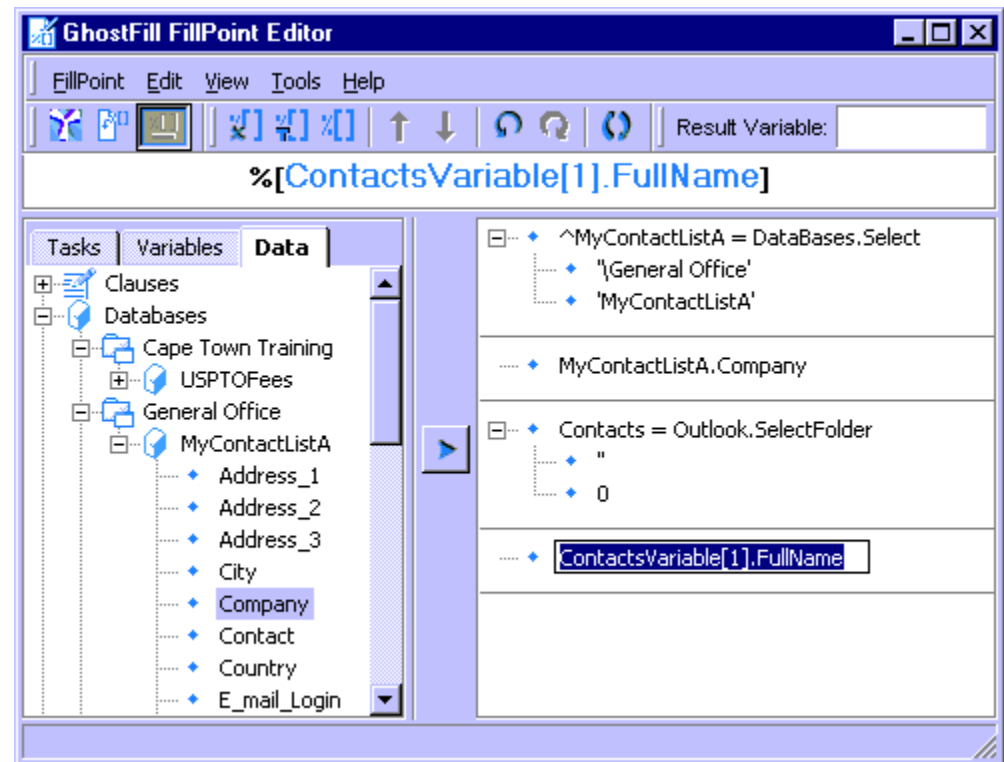
# Head to Head

## *External Data Sources*



### DataLink Wizard (ODBC)

- Name the DataLink
- Select/create data source
- Name the data source
- Select the data type
- Find the data
- Select the table or sheet
- Choose display columns
- Set access options
- DataLink Summary
- Ready to Use



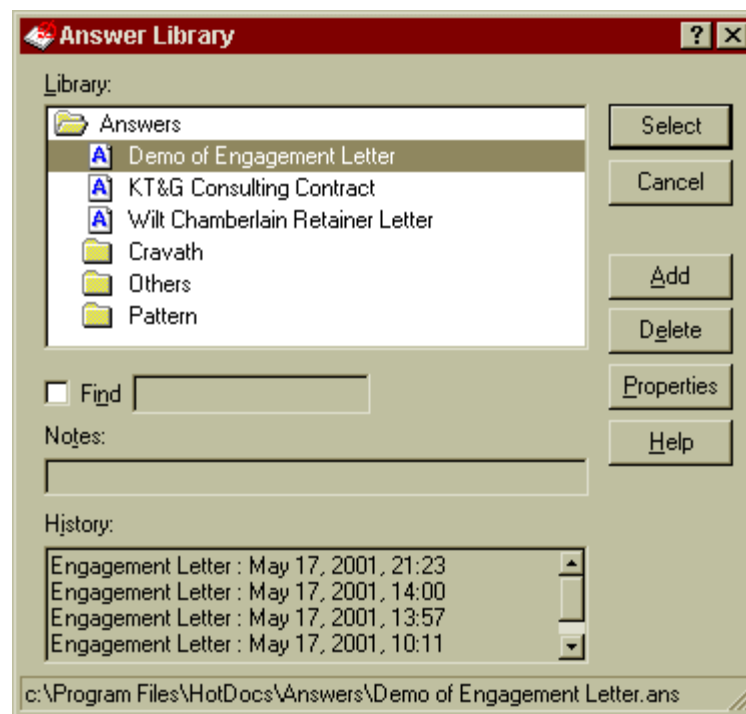


# Head to Head

## *Answer Files & XML*



- Answer File Options
  - Individual answer library
    - Stored on local drive
    - Cannot be effectively shared over a network
  - Answer file management off, with shared network directory
  - Answer file profiled and stored in iManage or DocsOpen





# Head to Head

## Answer Files & XML



- What is XML?
  - XML = Extensible Markup Language
- Why use XML?
  - Standard format to transfer data
  - HotDocs 5.3 & Online
- XML Definitions
  - Answers
  - Repeats
  - Variable types
  - Answer set title

```
Document Type Definitions

<!--ELEMENT AnswerSet (Answer*)-->
<!--ELEMENT Answer (TextValue | NumValue | DateValue | TFValue | MCValue |
ClauseLibValue | DBValue | RptValue)-->
<!--ELEMENT RptValue (TextValue* | NumValue* | DateValue* | TFValue |
MCValue* | ClauseLibValue* | DBValue* | RptValue*)-->

<!--ELEMENT TextValue (#PCDATA) >
<!--ELEMENT NumValue (#PCDATA) >
<!--ELEMENT DateValue (#PCDATA) >
<!--ELEMENT TFValue (#PCDATA) >
<!--ELEMENT MCValue (SelValue*) >
<!--ELEMENT ClauseLibValue (ClauseSelValue*) >
<!--ELEMENT DBValue (DBCColumn*) >

<!--ELEMENT DBCColumn (TextValue | NumValue | DateValue | TFValue | MCValue)-->
<!--ELEMENT SelValue (#PCDATA) >
<!--ELEMENT ClauseSelValue EMPTY >

<!--ATTLIST AnswerSet title CDATA #IMPLIED save (true | false | nochange)
#IMPLIED useMangledNames (true | false) #IMPLIED-->

<!--ATTLIST Answer name CDATA #REQUIRED save (true|false|nochange)#IMPLIED-->

<!--ATTLIST TextValue unans (true | false) #IMPLIED-->
<!--ATTLIST NumValue unans (true | false) #IMPLIED-->
<!--ATTLIST DateValue unans (true | false) #IMPLIED-->
<!--ATTLIST TFValue unans (true | false) #IMPLIED-->
<!--ATTLIST MCValue unans (true | false) #IMPLIED-->
<!--ATTLIST ClauseLibValue unans (true | false) #IMPLIED-->
<!--ATTLIST ClauseSelValue title CDATA #REQUIRED fileName CDATA #REQUIRED
description CDATA #IMPLIED-->
<!--ATTLIST DBCColumn name CDATA #REQUIRED-->
```



# Head to Head

## Answer Files & XML



- XML answers
  - AnswerSet title
  - Answer name
  - TFValue
  - DateValue
  - NumValue
  - TextValue
  - RPTValue
  - MCValue

```
<Answer name = "Simple project hourly rate NU">
  <NumValue>150</NumValue>
</Answer>
<Answer name = "(ANSWER FILE DESCRIPTION)">
  <TextValue></TextValue>
</Answer>
<Answer name = "Contractor TE">
  <TextValue>Weil</TextValue>
</Answer>
<Answer name = "Contractor city TE">
  <TextValue>New York</TextValue>
</Answer>
<Answer name = "Contractor name TE">
  <TextValue>Weil, Gotshal & Manges, LLP</TextValue>
</Answer>
<Answer name = "Contractor state TE">
  <TextValue>NY</TextValue>
</Answer>
<Answer name = "Contractor street1 TE">
  <TextValue>767 Fifth Avenue</TextValue>
</Answer>
<Answer name = "Contractor zip TE">
  <TextValue>10153</TextValue>
</Answer>
<Answer name = "Simple Project description TE">
  <TextValue>Automation of Generic Opinion Letter</TextValue>
</Answer>
<Answer name = "Simple project document TE">
  <RptValue>
    <TextValue>Opinion Letter</TextValue>
    <TextValue>Backup Memo</TextValue>
  </RptValue>
</Answer>
</AnswerSet>
```



# Head to Head

## Answer Files & XML



- Dialog answers stored in XML answer-sets
- Answer sets are editable independent of any document or template with a single click
- Persistent storage of *all* application data objects in XML:
  - Dialogs
  - Answer-sets
  - Datalinks
  - Clauses
  - Application mapping

The screenshot displays the GhostFill Explorer - Designer View interface. The main window shows a table with columns 'Name' and 'Modified'. The table contains one entry: 'FESmith' with a modified date of '6/3/01 9:31:18 PM'. Below the table, there is a 'Data' tab and a tree view showing a hierarchy of data objects. The tree view is expanded to show a list of fields: Full Name, Gender, Address Line 1, Town, County, State, Zip Code, Married?, Spouse's Full Name, Children?, Executor of Estate, Gender of executor, Alternative Executor, General Property details, Remainder of Estate details, Guardianship Details, Real Estate details, Specific Property details, and Date of signing. To the right of the tree view, there is a 'Dialog Name: SimpleWill' and 'Answer-Set: FESmith' section. Below this, there is a form with fields for: Full Name (Frederick Edward Srr), Gender (Male), Address Line 1 (21 Alberts Avenue), Town (Miami), County (Dale), State (Florida), Zip Code (12345), and a checked checkbox for Married?.



# Head to Head Use of XML



XML Data lends itself to multiple views

Raw XML

```
- <object clas
- <property
  Name" de
  <![CDATA
</propert
- <property
  displayna
  listname="Gender" depcnd="2" depfld="" depval="">
  <![CDATA[ Male ]]>
</property>
```

Basic View

GhostFill Dialog Structure

XML View XSL View

**GhostFill Dialog**

Basic View

Expand Entire Tree

Contract Entire Tree

Report View

GhostFill Dialog Structure

XML View XSL View Print

**GhostFill Dialog Structure**

Dialog Name: FESmith  
Dialog Class: \Law Office\Wills\Sir

Dialog Type: Embedded

No.	Field Name	Data Type	Prompt
1	Benefactor	String	Full Name
2	Gender	Stringlist	Gender
3	Address	String	Address Line 1

Field: Address

Dialog Summary

GhostFill Dialog Summary

Info Print

Dialog Name : SimpleWill  
Answer-set : FESmith

- FESmith
  - Benefactor [Frederick Edward Smith]
  - Gender [Male]
  - Address [21 Alberts Avenue]
  - Town [Miami]
  - County [Dale]
  - State [Florida]
  - ZipCode [12345]
  - Married [-1]
  - Spouse [Janet Smith]
  - Children [-1]
  - ExecutorofEstate [Frederick Jones]
  - ExecGender [Male]
  - AlternativeExecutor [Angela Jones]
  - GeneralPropertydetails
    - GeneralBeneficiary [all to my Children]
    - Other []

Total Dialogs : 6      Total Fields : 27  
Linked Sub-Dialogs : 0      Answered Fields : 24  
Embedded Sub-Dialogs : 6      Unanswered Fields : 3

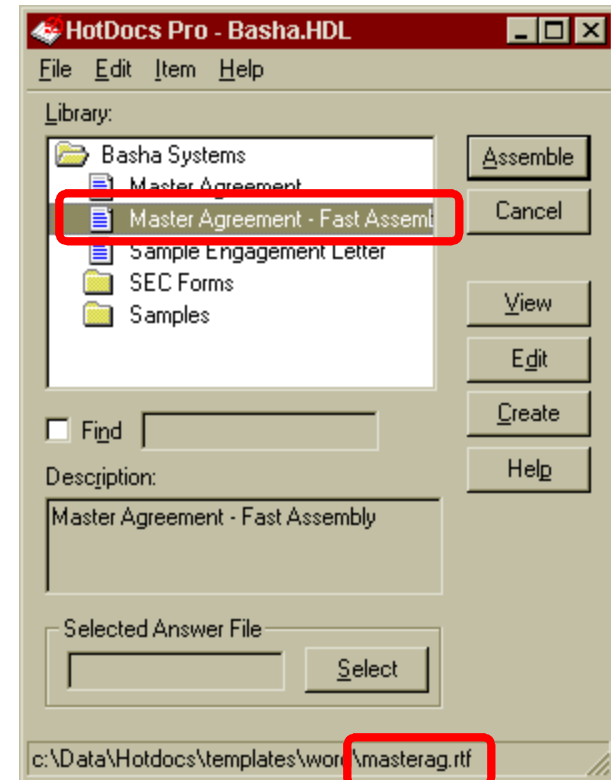


# Head to Head

## *Fast Assembly*



- Why Fast Assemble?
  - Long document assembled in seconds vs. minutes
  - DOT templates rely on slow Word macros for assembly
  - RTF templates don't
- Converting DOT to RTF
  - Select the template
  - Click CREATE
  - Change file extension & title
  - Click OK





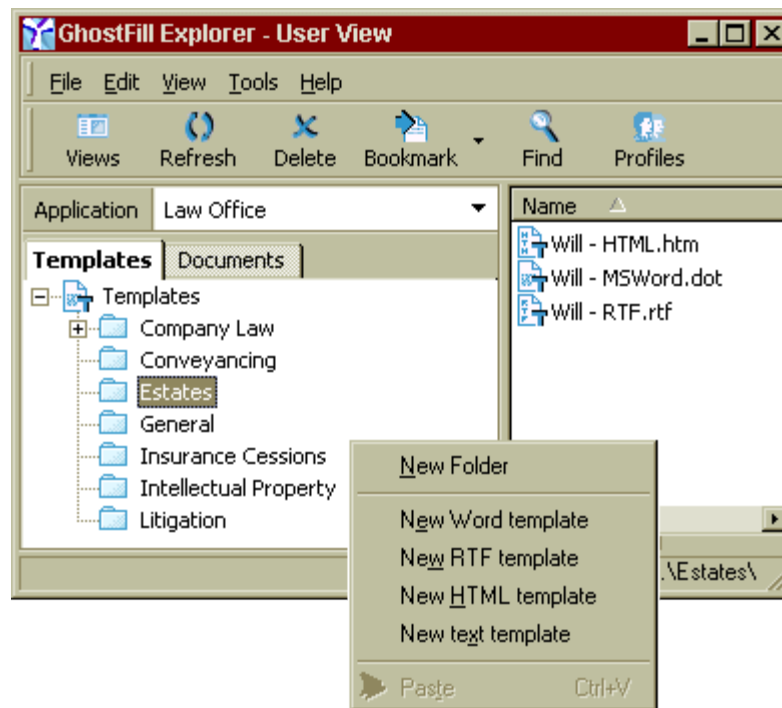


# Head to Head

## *Fast Assembly*



- Fill to Word uses Word to perform assembly
- Fill to RTF, HTML and plain text occurs outside word processor
- Can fill from GhostFill Explorer or via ActiveX control from another application
- Conversion ... just change the file format





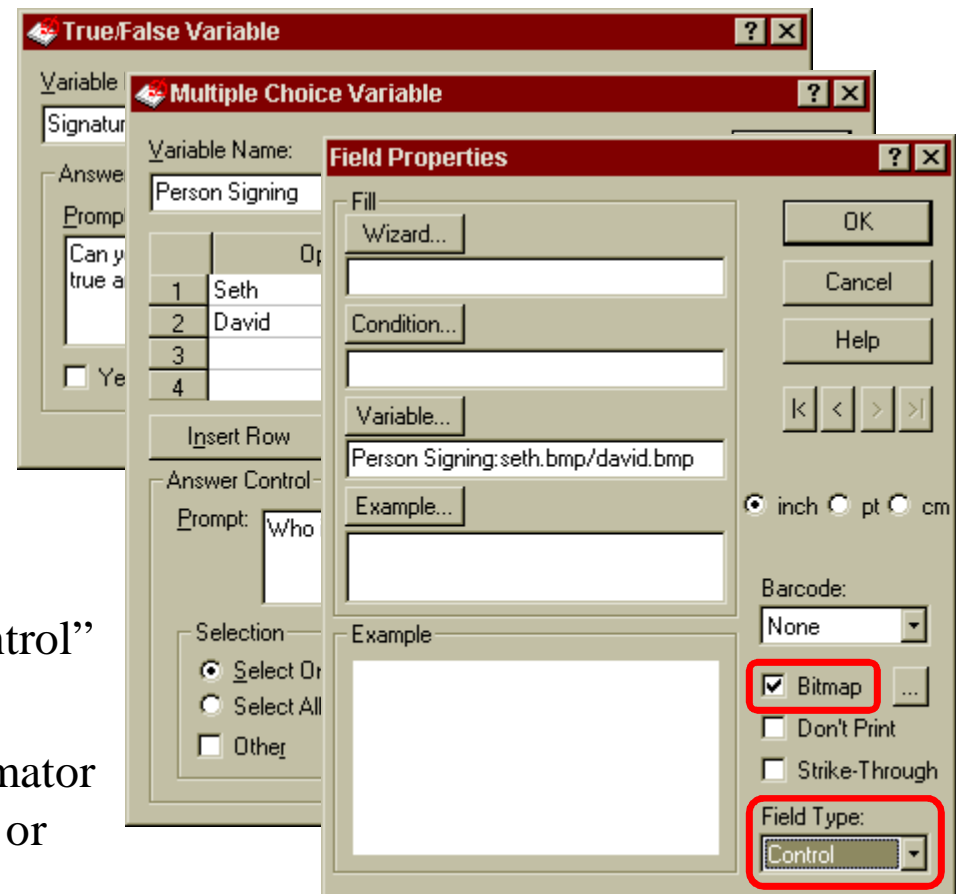
# Head to Head

## Rule-based Graphics



### ● Signatures

- For True/False variables, use a graphic file as your “Format Example”
- For multiple choice variables, use a graphic file as your “Merge Text”
- Define your Field Properties to allow bitmap files
- Set your “Field Type” to “Control” to make bitmap non-editable
- Caution: Limited to HD Automator ... else use INSERT Template or insert clause





# Head to Head

## Rule-based Graphics



### Signatures

- Define a new signature clause in GhostFill Explorer
- Drop your signature graphic into the clause
- Pull up a list of user profiles
- Reference the signature clause in the profile
- Place the variable anywhere in the document
- File Formats:
  - Bitmaps (bmp)
  - Metafiles (wmf)

The image displays three overlapping screenshots from the GhostFill software interface:

- Top Screenshot (GhostFill Explorer - Designer View):** Shows a table with columns 'Name' and 'Size'. A row is highlighted with 'AISmith' and '7 KB'. The 'Application' dropdown is set to 'General Office'.
- Middle Screenshot (GhostFill Clause Editor):** Shows a 'Clause' tab with a handwritten signature 'ASmith' on a white background. The 'Clauses' tree on the left shows a hierarchy: 'General Office' > 'Signatures' > 'AISmith'.
- Bottom Screenshot (GhostFill Explorer - Designer View):** Shows a table with columns 'Name' and 'Size'. A list of profiles is displayed:

Name	Size
AIS	1 KB
CMR	1 KB
FJW	1 KB
JanetJ	1 KB
JuneP	1 KB
MarkL	1 KB
- Bottom Screenshot (GhostFill - Profile Properties):** Shows a table with columns 'Variable Name' and 'Variable Value':

Variable Name	Variable Value
FeeEarnerInitials	AIS
FeeEarnerSignature	AISmith
FeeEarnerFullName	Adam I Smith
FeeEarnerEmail	asmith@wrs.com
FeeEarnerDirectPhone	658-9701



# Head to Head

## *Dynamic Merge Lists*



- Merge Text components

- Standard – Defined in Template Variable
- Merge list as a component that can be used for multiple variables
- Dealing with NEW items
  - In the Template
  - In the Merge Text
- Ability to add variables to text of merge list

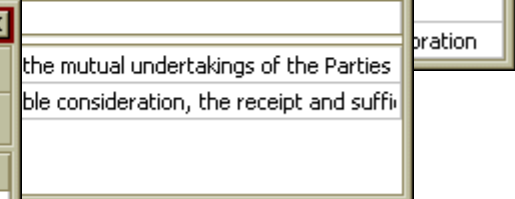
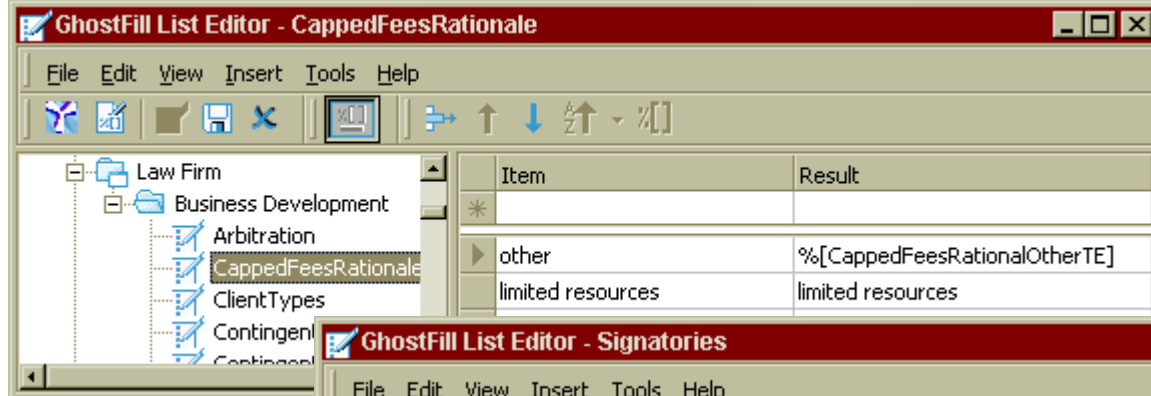
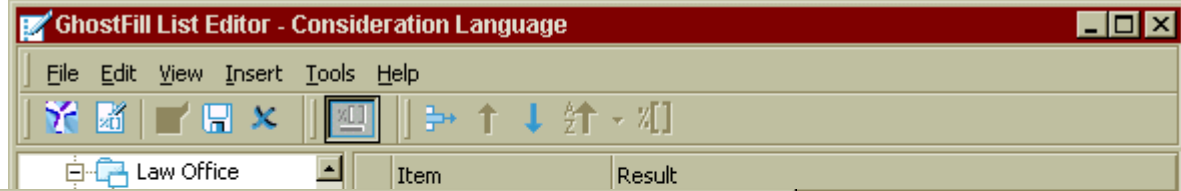
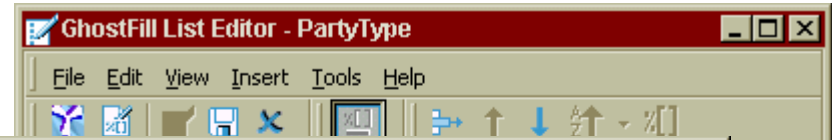


# Head to Head

## Dynamic Merge Lists

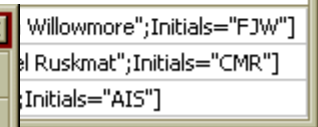
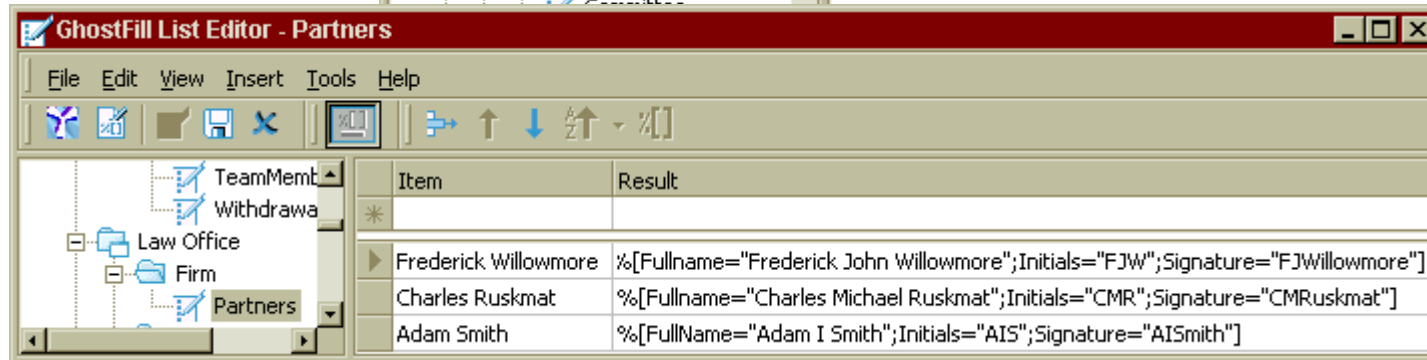


- List is an independent object
- Standard – Item matches Result
- Result inserts a pre-defined language



- One option calls a variable

- List sets value of two variables



- Bring in a clause (text, rules, bmp)



# Open Forum for Questions



**CONTACT INFO:**  
**Seth G. Rowland, Esq.**  
CEO & President  
Basha Systems LLC  
[sgr@bashasys.com](mailto:sgr@bashasys.com)  
[www.bashasys.com](http://www.bashasys.com)





# GhostFill Training

- Introduction to GhostFill 2001
- Fillpoints and Functions
  - Types of Fillpoints
  - Working with Objects
- External Data Sources
  - Outlook
  - Excel/Access/SQL
- Dialog Design
- Lists and Profiles
- GhostFill Back Office

